Towards Better Character-based Word Vectors

Luoshu Wang
luoshu@stanford.edu

Abstract

Morphemes have proven to be important in understanding the meanings of words in many languages, e.g., English. They are also popular in the world of learning word representation. Here are some issues in current approaches. Character-based word vectors, such as [1], implicitly assume that the every morpheme is equally important regardless of the context, but, in reality, that is not the case. An effective way to identify which subwords are the most relevant remains to be found.

In this study, we propose a new model that represents words by only using the sum of important subwords. (1) We first design an unsupervised word segmentation model to split words into morphemes by combining byte-pair-encoding(BPE) method and word weights. We also propose a new method called byte-pair-search(BPS) to enrich subword information. (2) We propose representing words by only using the sum of these segmented morphemes, instead of all ngrams. (3) Finally, we conduct an extensive experimental study to demonstrate the effectiveness and efficiency of our approach, using wikipedia data.

1 Introduction

Word representation has drawn a lot of attention in the NLP community. It has been the key component of many NLP tasks, such as language modeling, machine translation, text classification and sentiment analysis. In recent years, a lot of work [2] [9] has been done to update word vectors by combining context information. However, more exploration is needed to improve word representation using its internal structure.

Some prior work has proven subword units are helpful for word representation. The FastText method [1] extends the skipgram and negative sampling method [6] and proposes representing words by taking into account their ngrams. The FastText method implicitly assumes that every morpheme is equally important regardless of the context, but, in reality, that's not the case. [11] propose to use byte-pair-encoding in extracting important subword information, which has proven to be a successful attempt in neural machine translation.

In this study, we first extend the work of [1] and adopt an unsupervised word segmentation model to split words into morphemes by combining byte-pair-encoding(BPE) method and word weights. We also propose a new method called byte-pair-search(BPS) to enrich subword information. We think BPE might lose the important morphemes since it only chooses one most frequent pair in every iteration even there are multiple candidates, and the merge order might influence the final word segmentation results. Next we feed our Subword Model back to the general models. We inherited the skipgram model and negative sampling method as our general model framework. For speeding up building the models, we also propose a subword logistic regression model to quick fit our subword model to pretrained word vectors. Finally, we evaluate our model on human similarity judgment tasks, using FastText model as the baseline. We also analyzed the impact of word weight, vocab size, Begin-Of-Word and End-Of-Word in word representation.

2 Related Work

Distributed word representation has been widely studied. In 2013, [6] introduced the word2vec method which allows to train word embedding through shallow neural network using word co-occurrence information. In 2014, GloVe model [8] was proposed to leverage statistical information from both global matrix factorization and local context window. In 2018, BERT [2] and ELMo [9] used context information to update original word vector and got exceptional results.

Besides context knowledge, using subword information as NLP feature has also been recently investigated. In 2013, [7] added knowledge-based morphological features to word representation. In 2014, [10] proposed jointly learning subword units and word to enrich word representation. In 2016, [1] extended the work of word2vec [6] and proposed the FastText method, which representing word using a bag of ngrams. [12] proposed wordpiece model and used byte-pair-encoding method [11] to deal with rare words in neural machine translation. In 2018, the BERT model [2] inherited the usage of wordpiece model. In our model, we also extend the work of byte-pair-encoding, but the major difference is that [12] [2] focus on word piece representation and our model focus on word representation itself.

3 Approach

Our data-driven model learns word representations while taking into account morphology. In this section, we first present the basic concepts that are defined and used in our model. Then we present our Subword Model and introduce how to segment words using byte-pair-encoding method and word weight assumption. Next we talk about how to represent words by only using the segmented subwords. Finally, we show the general training models.

3.1 Subword Model

We first present the key characteristics of our Subword Model.

Word weight. In the input representation of our model, every word w has its own weight weight(w), which indicates the importance of each word across all words in vocabulary. We propose to represent word weight associating its actual frequency in articles, say:

$$weight(w) = freq(word)^{\alpha} \tag{1}$$

α is a hyperparameter, we recommend to use value between [0.75, 1], which performs best in the experiments.

Byte-pair-encoding (BPE). When looking at words and their word weights, we can adopt the byte-pair-encoding methods to segment words, which is similar to the usage in [11] to deal with rare words in Neural Machine Translation.

Byte-pair-search (BPS). We extend the work of BPE, and propose a new model called Byte-pair-search to select the most important subwords and enrich the representation. In every iteration in the BPE approach, BPE chooses the most frequent subword pair and merges them into a new subword unit. We argue that when two subword pairs havethe same frequency occurrence, the merge order might influence the final word segmentation results, which might lead to the loss of important morphemes. In our approach, every time when the model chooses most frequent subword pair, if there are more than one candidate, we will keep all of them. Our BPS approach can efficiently help with addressing this issue by going through all the possibilities and returning more relevant subwords.

Subword frequency. BPE chooses the most frequent subword pair in every iteration and merges it into a new subword unit. Freq(AB) is the frequency of A and B occurring together as a one word unit. Subword frequency is a key threshold to achieve a balance between keeping high-level words in the vocabulary and effective splitting.

Example 1: For the word 'understanding', in our model we will split the word into

understanding: understand ing

If we use a lower word frequency threshold, we might split 'understanding' into 'under', 'stand' and 'ing', or even lower-level n-grams 'st' and 'and', which does not take 'understand' as a whole subword. If we use a higher word frequency threshold, we might leave most words unsegmented.

BOW and EOW. Begin-Of-Word(BOW) and End-Of-Word(EOW) play an important role in word representation, and we study their impact in our model. We use '<' as the identifier for the BOW and use '>' as the identifier of EOW, since there is no word that has both '<' and '>' in its origin vocabulary.

Example 2: <cleanup> is a word with BOW and EOW.

Words with BOW and EOW have some interesting patterns.

Fact 1: Freq(<word>) always equals 1 if there is no other word that contains <word>.

Fact 2: All words are split into two or more subwords in the final step if learning BPE using subword frequency cutoff is larger than 1 and there is no other word that contains <word>.

Intuitively, Fact 1 and Fact 2 tell us if we want to use word representation with both BOW and EOW, we had better add origin <word> back to vocabulary. Otherwise, we will lose the insight of taking <word> as a whole.

If representing input words without BOW and EOW or only having BOW or EOW, the model could benefit from taking its original word as a subword of other words naturally and learn one word prefixes or suffixes as subwords of another word, which is especially useful for long word representation. We prefer using this approach in our model.

Word Segmentation Model. We then present the word segmentation model for both common words and out-of-vocabulary words. We can learn the most frequent subword pairs using BPE or BPS method from a dictionary, e.g., wikipedia articles, and then sort these subword pairs by frequency. This pre-training method allows us to segment words using large dictionary information, instead of depending on training data, which might be small. While getting common words segmentation results in BPE, we are also able to segment Out-of-vocabulary words efficiently as described in following algorithm. We set up minimum word frequency threshold to control how deep we would like to segment words.

```
def oov_word_segment(word, most_freq_pairs, min_frequency):
    word_segment_res = list(word)
    for most_freq_pair in most_freq_pairs:
        if (most_freq_pair.frequency < min_frequency):
            break
        # If there is merge candidate.
        if (contains_pair(word_segment_res, most_freq_pair)):
            # Merge two subwords, e.g., abcd -> AEd
            update_word_segment(word_segment_res, most_freq_pair)
    return word_segment_res
```

3.2 Representing words by segmented subwords

After getting segmented subwords, we can represent words by only using the segmented results. The traditional character-based word vectors, such as [1], attempts to represent words by ngrams.

Example 3: Take the word <where> and n = 3 as an example. The word will be represented by the sum of ngrams in the FastText model: <wh, whe, her, ere, re> and the special sequence <where>.

However, in this case, we think some subwords, e.g., her, is useless in representing <where>. Since the subword is irrelevant to the original word while taking into account human knowledge. We only want to represent words using meaningful ngrams.

Model We propose a model that represents words only using the segmented subwords. Given a dictionary of segmented subwords S and a word W, we will represent a word by the sum of the vector representations of its segmented subwords built in section 1.1. The scoring function is:

$$s(w, c) = \sum_{s \in S} z_s^T v_c \tag{2}$$

Example 4: Using the word 'precise' as an example, the segmented subword will be prec and ise. We propose to present this word as

s('precise') = s('prec') + s('ise')

3.3 General Model

After obtaining word segmentation information, we can feed the subword model back to FastText [1] embedding learning framework, which was built on the top of skip-gram model and negative sampling methods [6]. We need to modify the ngrams representation part and OOV word segmentation to serve our subword models.

Another way to learn word representation is using subword segmentation results to fit with a pretrained original vector using a simple logistic regression model. This approach can help accelerate the speed of building word vectors. The loss function would be:

$$loss = sum(embeddings(subwords)) - embedding(word) \tag{3}$$

Example 5: word could be segmented to three subwords: A, B, C . We will have a loss function:

loss = embedding(A) + embedding(B) + embedding(C) - pretrained embedding(word)

We can train model, minimize the loss and then obtain the character-based word vectors.

4 Experiments

4.1 Baseline

In this study, we compare our model to the C++ implementaion of skipgram models from FastText package [1].

4.2 Data

We build vocabulary using both enwik9 dataset and Wiki latest pages articles. We train embeddings in enwik9 data. For comparison to prior work [1], we also use Matt Mahoney's pre-processing perl script to shuffle the data. We use the WordSim353 dataset introduced by [3] and the Rare Word dataset introduced by [5] to do human similarity model evaluation. For comparison to previous work [4], our model can be evaluated in text classification tasks as well. We also evaluate in text classification tasks using 8 datasets and evaluation protocol of [13].

4.3 Experimental setup

To compare with [1], we adopt the same experimental setup as baseline. The word vectors have dimension 300, and we randomly sample 5 negatives for each positive example. The learning rate is 0.05 for skip-gram model and epoch is 5. Only the words that appear at least 5 times in the training set will be kept in the word dictionary. The evaluation model is set up in C++. The human similarity judgement taks are conducted with BPE-based Subword Model.

4.4 Results

Human similarity judgment tasks We evaluate our BPE-based model on the word similarity tasks, by computing Spearman's rank correlation coefficient between human judgment and the cosine

similarity between the vector representations. We use both WordSim353 dataset and Rare Word dataset.

From the Table 1, our model outperforms FastText in WordSim353 task and less effective in Rare Word task. The reason behind is that there is more common words in WordSim353 dataset, where our model might be benefited from the morphological decomposition of words. The loss in the Rare Word dataset might come from BPE can only split word to non-duplicated subwords, so if the segmentation is imperfect, it might lose important information. BPS method might help with this case.

Table 1: Human Similarity Judgment

Model	**WS353**	**Rare Word**
FastText (enwik9 as dictionary)	74	45
Our Model (enwik9 as dictionary)	76	41

Effect of Dictionary Size We build word segmentation dictionary from both enwik9 dataset(713M, 21K words after filtering) and Wiki latest pages articles(22G, 1.7M words after filtering). From the experiments, even using enwik9 as dictionary, our model is able to segment all OOV words efficiently and get the similar performance as super large dictionary.

Table 2: Dictionary Size / Human Similarity Judgment

Model	**WS353**	**Rare Word**
Our Model (enwik9 as dictionary)	76	41
Our Model (Wiki latest pages articles as dictionary)	76	41

4.5 Qualitative analysis

Word segmentation results We report sample BPE-based word segmentation results in Table3, which are randomly selected. From the results, we are able to capture the correct morphemes within one word. Our models outperforms FastText when the morpheme is a long subword, e.g. 'transfer', which is hard to capture by FastText model since by default FastText only catch ngrams with length range [3, 6].

Table 3: Word segmentation result / Same morpheme

Word	**Segmentation results**
transferring	transfer ring
transferable	transfer able
transference	transfer ence
transferred	transfer red
deliveryman	delivery man
extraordinarius	extra ordin ar ius
dissociatives	dis soci atives
childishness	child ish ness
farmerville	farmer ville

Compare BPS and BPE When we study the impact of BPS and BPE, we find BPS can help enrich our vocabulary. Use 'low 5, lower 2, newest 6, widest 3' as an input example and set the subword frequency as 5. Table 4 is the vocaulary generated by BPE and BPS method. We have implemented the basic version of BPS and plan to add optimization methods to speed up the pre-training. Please refer future work section for more details.

5 Conclusion and Future Work

Conclusion This project is attempt to build a word representation models using subword information. We first develop a subword segmentation model, which is an extension work to byte pair

Table 4: Vocabulary from BPE and BPS

BPE	**BPS**
s t st 9	es t est 9
e st est 9	e st est 9
o w ow 7	s t st 9
l ow low 7	l o lo 7
w est west 6	o w ow 7
n e ne 6	lo w low 7
ne west newest 6	new est newest 6
N/A	e west ewest 6
N/A	ne w new 6
N/A	e w ew 6
N/A	w est west 6
N/A	n e ne 6
N/A	ew est ewest 6
N/A	ne west newest 6
N/A	n ew new 6

encoding. And then we propose to represent word by only using these segmented results. We feed them back to the general training framework and get the word embedding. We evaluate our word vector and it outperforms FastText baseline in WordSim353 dataset in the Human Similarity Judgment tasks. Our model allows training embedding very fast, which is around 2x faster than baseline in enwik9 dataset while capturing morphones within one word effectively.

Future work In the next a few weeks, we are going to implement an efficient version of byte-pair-search(BPS) to select the most important subwords and enrich the representation. We can add optimization methods such as multithreading and removing duplication to speed up the pre-training process. We will also run word embedding training in larger wikipedia dataset rather than enwik9 to analyze the model performance.

Acknowledgments

Huge thanks to Peng Qi for the mentorship on this project!

References

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[3] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131, 2002.

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[5] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, 2013.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[7] Amr El-Desoky Mousa, Hong-Kwang Jeff Kuo, Lidia Mangu, and Hagen Soltau. Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8435–8439. IEEE, 2013.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[10] Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 141–150, 2014.

[11] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015.

[12] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[13] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.